On the Use of Variability Operations in the V-Modell XT Software Process Line

Kuhrmann, Marco; Méndez Fernández, Daniel; Ternité, Thomas

Published in:
Journal of Software: Evolution and Process

DOI:
10.1002/smr.1751

Publication date:
2016

Document version
Submitted manuscript

Document license
Unspecified

Citation for pulished version (APA):
Kuhrmann, M., Méndez Fernández, D., & Ternité, T. (2016). On the Use of Variability Operations in the V-Modell XT Software Process Line. Journal of Software: Evolution and Process, 28(4), 241-253. DOI: 10.1002/smr.1751

On the Use of Variability Operations in the V-Modell XT Software Process Line

Marco Kuhrmann[1], Daniel Méndez Fernández[2], Thomas Ternité[3]

[1]University of Southern Denmark, Mærsk Mc-Kinney Møller Institute & Center for Energy Informatics, Campusvej 55, DK-5230 Odense M, Denmark

[2]Technische Universität München, Faculty of Informatics – Software & Systems Engineering, Boltzmannstr. 3 85748 Garching, Germany

[3]Technische Universität Clausthal, Depatment of Informatics, Clausthal-Zellerfeld, Germany

Corresponding Contact:
E-Mail: kuhrmann@mmmi.sdu.dk
Phone: +45 24 60 14 22

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
J. Softw. Evol. and Proc. 0000; **00**:1–12
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr

On the Use of Variability Operations in the V-Modell XT Software Process Line

Marco Kuhrmann[1*], Daniel Méndez Fernández[2], Thomas Ternité[3]

[1]*University of Southern Denmark, The Mærsk Mc-Kinney Møller Institute & The Center for Energy Informatics, Odense, Denmark*
[2]*Technische Universität München, Faculty of Informatics, Garching, Germany*
[3]*Technische Universität Clausthal, Department of Informatics, Clausthal-Zellerfeld, Germany*

SUMMARY

Software process lines provide a systematic approach to develop and manage software processes. A process line defines a reference process containing general process assets, whereas a well-defined customization approach allows process engineers to create new process variants, e.g., by extending or modifying process assets. Variability operations are an instrument to realize flexibility in the V-Modell XT process line by explicitly declaring required modifications, which are applied in a later step to create a procedurally generated company-specific process. However, little is yet known about which variability operations are suitable in practice. In this article, we present a study on the feasibility of variability operations to support the development of software process lines in the context of the German V-Modell XT. We analyze which variability operations are defined and practically used, and if not used, why. We provide an initial catalog of variability operations as an improvement proposal for other process models. Our findings show that 69 variability operation types are defined across several metamodel versions of which, however, 25 remain unused. The found variability operations allow for systematically modifying the content of process model elements and the process documentation, and they allow for altering the structure of a process model and its description. Furthermore, we also find that variability operations can help process engineers to compensate process metamodel evolution. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Software Process Lines; Metamodel Evolution; Variability Operations

1. INTRODUCTION

The V-Modell is the standard software process model for public sector IT development projects in Germany. It has a long history beginning with its first release in 1992, and it was improved in several iterations. Since 2005, the revised version of the V-Modell XT became subject to numerous adaptations, which were, initially, conducted following a "copy & change" procedure in which company-specific process variants were realized by directly modifying a local copy of the reference process. As the reference process evolved [10], this approach caused serious problems, e.g., when integrating updated contents, figuring out what particular customizations were affected by newer reference contents, and when migrating existing content to a new process metamodel. Much effort has been spent to analyze the evolved variants (see, e.g., [18–20]). However, only the changes could be analyzed and documented. Efficiently integrating evolved model contents with customized ones to create a new version of the company-specific process remained a critical and unresolved task.

*Correspondence to: University of Southern Denmark, The Mærsk Mc-Kinney Møller Institute & The Center for Energy Informatics, Campusvej 55, 5230 Odense M, Denmark. E-Mail: kuhrmann@mmmi.sdu.dk

In response to this problem, in the WEIT[†] project, it was decided to adopt principles from *Software Process Lines* (SPrL, [23]) to maintain the reference process *and* its variants to allow for evolution and (automatic) updates. Special attention was paid to the customization[‡] approach to support process engineers in, among other things, using *typed variability operations* [26] as a declarative instrument to systematically derive a company-specific process variant from a reference model while ensuring consistency and addressing compliance requirements.

Problem Statement While defining the variability operations for the V-Modell XT, a major problem was to define a set of suitable variability operations. Available approaches are either of conceptual nature [17] or they focus on general concepts [22] and require further refinement. That is, process engineers need to develop their own portfolio of variability instruments negatively impacting the SPrL, e.g., due to incompatible sets of variability operations, and potential losses of compliance. Missing is a set of actionable variability operations to support process variant development within a comprehensive SPrL. Furthermore, we still lack long-term studies analyzing the feasibility of SPrL approaches [5, 7].

Contribution In this article, we contribute an exploratory study on the application of variability operations to realize a comprehensive SPrL. We study a snapshot from 2013 of the V-Modell XT process line [10] in which we investigated the V-Modell XT reference model and 5 of its variants using the built-in SPrL features. We contribute an initial catalog of variability operations as implemented in the V-Modell XT [11], and analyze the feasibility of this instrument. We investigate which variability operations were defined and to what extent those were used in practice. Furthermore, we analyze settings in which variability operations were not used and provide a rationale. In addition to our previously published study [12], we provide a broader perspective on the context of our contribution and provide more details on the variability operation instrument.

Outline The remainder of this article is organized as follows: Section 2 introduces the basic concepts and terminology used and summarizes the related work. In Section 3, we present the study design including our research questions, the case selection, and the data collection and analysis procedures. In Section 4, we finally present our findings, before giving a conclusion in Section 5.

2. FUNDAMENTALS & RELATED WORK

In this section, we provide a brief overview of the key concepts behind the V-Modell XT process line, and we discuss related work.

Context & Used Terminology We analyzed the V-Modell XT SPrL [10], which is a process framework (including, e.g., metamodels, tools, reference implementations, and guidelines) with built-in SPrL features. In order to provide understanding of variability operations as one of these features, in Table I, we introduce the basic concepts and underlying terminology.

[†] **WE**iterentwicklung des **IT**-Entwicklungsstandards des Bundes—SPI project to improve the standard IT development process model of the German government.

[‡] The V-Modell XT supports process customization at different levels: At the organization level, companies can develop a company-specific software process variant, which can be grounded in a reference model. Furthermore, the V-Modell XT supports the project-specific tailoring in which a (company-specific) process is tailored according to the actual project context. This article addresses the organization level and presents an instrument used to develop a process variant from a reference process in the context of a process line. Referring to the nomenclature from [24] (PD: Process Description, MM: Metamodel), this article covers the areas PDMM, PD, Adapt PDMM, and Create/Update PD. The article addresses process engineers (PE) rather than project managers.

DOI: 10.1002/smr

Table I. Basic terminology and concepts in the V-Modell XT process line.

Concept	Description
Model, Metamodel and Modules	The V-Modell XT is a modular, metamodel-based framework to define software processes. The metamodel [28] defines the *process language* to create single processes and SPrLs. A *V-Modell-variant* logically consist of two models: (1) a structure model contains all (atomic) model elements, and (2) an overlaying dependency model connects all model elements. Hence, if dependencies are contained in *process modules*, the configuration of such modules directly influences the dependency model, which allows for a comprehensive tailoring [9]. During customization, all elements of these packages can be extended, altered, and so on.
Process Variants	The metamodel supports hierarchically organized process variants—even the reference model is seen as a variant [26, 27]. Creating a new variant requires a *reference model* on which the variant is based. A variant can be regarded as an *extension* applied to a reference model. Since all V-Modell-variants are (or should be) based on the same metamodel, each variant may contain a complete process description. As all model elements from the reference model are accessible from a variant, a variant can refer to and, thus, integrate and modify any reference model element. A merge tool creates an integrated process description from the variants. New assets introduced by the variant will then be integrated with the reference model, exclusions will be deleted, and variability operations will be processed.
V-Modell XT Process Line	The V-Modell XT family consists of a reference process and a number of derived variants [10]. The snapshot, which is the case for our study (Figure 2), shows the reference model and two kinds of variants: We find variants created as a direct modification of a local copy ("old" scheme), and we find 5 variants using the built-in SPrL features. Variants using the SPrL features can reuse content from the reference model and support automatic updates. If a new version of the reference process is released, in the simple case, the merge tool automatically updates a variant, e.g., by computing the variability operations again.

Variability Operations As part of the customization framework, variability operations allow a process variant to modify contents and structure of the reference model [26]. A variability operation is a model element that declares a change, e.g., renaming of elements, adding description text, or restructuring dependencies (Figure 1). Furthermore, the instrument described in this article provides *typed* variability operations, i.e., the operations have a specific purpose. For example, if the task is to rename a role, the respective operation is named *RenameRole* and it only refers to elements of type *Role* in the process model. The complex variability operations—as provided to process engineers—are realized by assembling certain atomic model-transformation operations, such as *RenameElement*, *AddText*, *ReplaceText*, or *SwapRefences* [27]. From these atomic operations, if applicable, process engineers may also compose new operation types to extend the set of available operations in response to particular customer requirements (Section 4.2).

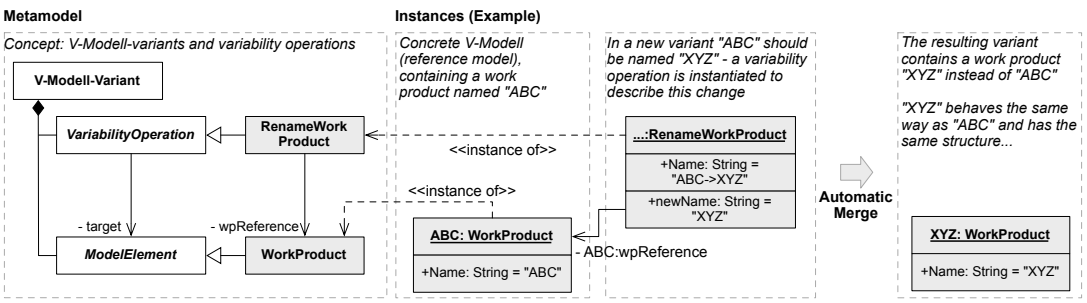

Figure 1. Variability operations (concept and example).

A particular variability operation exemplar refers to a model element in the reference process, and describes how the referred element will be treated during the merge procedure in which the reference model and an extension model are processed in order to compile the company-specific process variant (Table I), i.e., variability operations are used during the definition of a company-specific process. Utilizing variability operations ensures that the source files of the reference model are not touched, as all change declarations are contained in an externally managed extension model. During the merge procedure, the information provided by variability operations

DOI: 10.1002/smr

is operationalized by the merge tool. For example, the declaration of renaming a work product has to be evaluated and executed during that merge. If an extension model contains an exemplar of the *RenameWorkProduct* operation, this has to be interpreted as a rename operation (execute the atomic operation *RenameElement*) on the referred instance of 'WorkProduct' during the merge (Figure 1). After the merge procedure, the integrated company-specific process variant can be deployed to the projects in which the process is subject to further tailoring and, eventually, application.

Related Work In [23], Rombach votes for organizing comprehensive software processes similar to product lines. To this end, SPrLs consist of a stable core (commonalities) and variable parts (variabilities) [4, 6, 17, 25]. Software process lines propose advantages regarding the organization and management of process knowledge and the systematic creation of reusable process assets to ease process variant development. A software process line is a framework for a directed and proactive process construction and management. Yet only few publications deal with self-contained approaches providing support for process engineers. For instance, in [1], Alegría and Bastarrica present CASPER, which they define as meta-process to support the construction of project-specific processes. Martínez-Ruiz et al. [13] and Oliveira et al. [21] propose extensions of SPEM's variability constructors, and address the problem that SPEM, basically, provides generic variability operations and modularization concepts, but does not provide explicit and context-specific variability constructors. SPEM [22] and the V-Modell XT [28] explicitly define variability operations. Process assets built on these metamodels can extend or modify other process assets, and they can be configured from certain (process) modules. However, in contrast to SPEM that only provides a generic concept of variability, the V-Modell XT explicitly defines a process variant concept and provides an extensive set of *typed* variability operations for fine-grained model manipulations. In [27], Ternité describes typed variability operations to support software process variability (Figure 1). This instrument is used in the V-Modell XT to realize a complex software process line (Figure 2).

Instead of constructively defining process variants, Münch et al. focus on an evolutionary approach that comprises: (1) scoping processes to work out where variability is needed [2, 3] by determining the properties an actual process has and by identifying commonalities/pattern to infer needed variability; (2) providing rationale during process evolution [18–20]; and (3) analyzing differences of evolved model variants. This approach is based on an *a posteriori* observation of the evolved subject.

Although considered of particular interest, SPrL concepts are still considered immature due to a lack of empirical evidence for their feasibility [5, 7], or due to the absence of a meaningful notation for variable processes [15]. Also, a deeper understanding of the variability instruments is in general yet missing. With the study at hand, we close this gap in literature.

3. STUDY DESIGN

In this section, we present the study design. After defining the goal and the research questions, we describe how we selected the case. Finally, we describe how we collected and analyzed the data.

3.1. Research Questions

Our overall objective is to study the feasibility and the practical application of variability operations to support the (long-term) development and the maintenance of SPrLs. To this end, we investigate which variability operations are implemented in general and to what extend these operations are used. In a second step, we study settings in which variability operations were not used and why. We define our research questions in Table II.

DOI: 10.1002/smr

Table II. Research questions.

No.	Description and Rationale
RQ_1	*Which variability operations are defined to realize the process line?* Since most related work discusses—if at all—variability operations in a generic manner, our first research question aims to identify a set of variability operations to create a catalog.
RQ_2	*Which variability operations are practically used to which extent?* This question aims to investigate the feasibility of the found variability operations. We analyzed to which extent the found variability operation types were actually used in particular process variants.
RQ_3	*In which settings are variability operations not used and why?* We aim to investigate settings that are potentially inappropriate for variability operations. We studied settings in which variability operations were not used, and we investigated the respective settings, analyzed the instruments used instead of variability operations, and provide a rationale.

3.2. Case Selection

We opted for the V-Modell XT to collect and analyze variability operations. As we were interested in the variability operations and their use, we only considered such variants using the built-in SPrL features—all other variants are out of scope.

3.3. Data Collection Procedure

To answer RQ_1 and RQ_2, we used a tool to export lists of the variability operations defined and used. All information was collected by parsing the models' XML files and storing the data in a spreadsheet. Therefore, we first analyzed the respective metamodel on which a process variant is based and gathered all defined *variability operation types*. In a second step, we exported the *variability operation exemplars* as defined in the process models (in this step, we also analyzed which version of the metamodel defines an operation to track the metamodel evolution). We repeated the export process for all considered variants to create (1) a consolidated list of operation types across all versions of the metamodel, (2) process-variant-specific lists of variability operation exemplars, and (3) an aggregated list of all variability operations, their type, number of exemplars, and so forth. In order to answer RQ_3, we had to (manually) inspect the considered process variants. We compared the merged process definition with its sources for added, modified, and/or removed process assets that are not defined using variability operations. The outcome of this investigation was also stored in a spreadsheet.

3.4. Analysis Procedure

Due to the low number of cases, we present the results as data tables and simple charts, and qualitatively analyze and interpret the results.

4. STUDY RESULTS

We first give a description of the case, before summarizing the results structured according to the research questions.

4.1. Case Description

As case, we opt for the V-Modell XT and the set of 5 variants that use the built-in SPrL features. Figure 2 shows a snapshot; the highlighted variants are subject to the study—the other variants do not use the SPrL features and, thus, are out of scope. Our study is based on the V-Modell XT 1.3[§]

[§]Although the V-Modell XT 1.4 was released in 2012, no variants using this version as reference model were available when we conducted the study.

DOI: 10.1002/smr

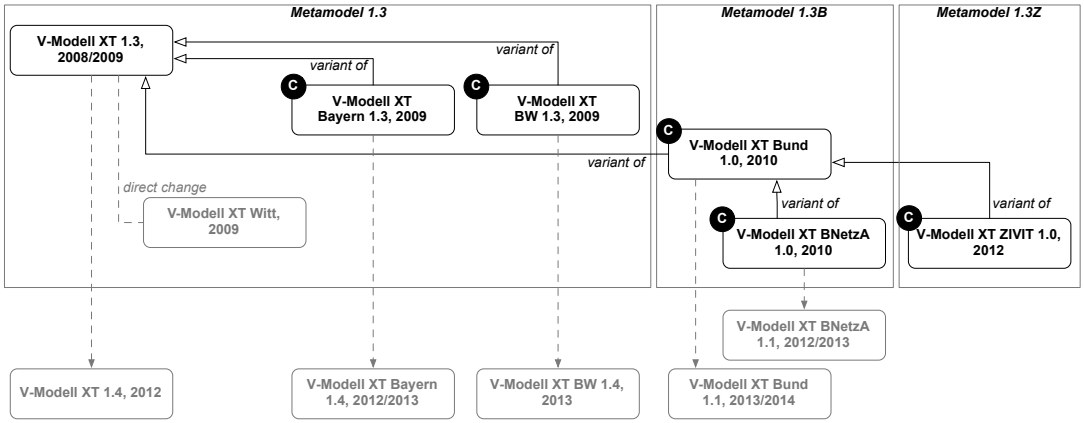

Figure 2. Snapshot of the V-Modell XT software process line (variants marked with "C" form our case).

and we refer to this version as the *reference model* on which, finally, all variants[¶] are built. Each variant, except for the V-Modell XT 1.3, points to its parent (Figure 2 also shows that the SPrL builds a "family tree" in which a derived variant can be a reference model for further variants).

4.2. RQ 1: Defined Variability Operation Types

Since the variants under consideration use different versions/variants of the metamodel (Figure 2), we first analyze (1) which variability operation types are defined and (2) which metamodel defines a particular operation type.

Due to space limitations, the complete list (the catalog) of variability operations is not part of this article, but is available for download [11]. In total, the V-Modell XT metamodel provides process engineers with 69 variability operation types, which are defined by the metamodel versions "1.3" and "1.3B" (the metamodel "1.3Z" does not define new operation types). Figure 3 summarizes the number of defined variability operation types per operation group[‖] and per defining metamodel version. Furthermore, Figure 3 also reflects the evolution of the metamodel—35 new operation types were introduced in the metamodel "1.3B" (two years after the publication of the reference model 1.3, which defines 34 variability operation types).

Interpretation We found variability operation types defined in two metamodel versions. Moreover, the number of operation types doubled. An explanation can be found in the metamodel's evolution. The metamodel "1.3B" got a substantial improvement, which was based on customer requirements, whereas the initial set of operation types was derived from known improvements at this time and compliance requirements in the context of a certification program. So far, the growing number of operation types indicates that the mechanism "variability operation" can be used to foster flexibility in a process line (in response to customer requirements).

4.3. RQ 2: Variability Operation Usage

The second research question aims at investigating which of the defined operation types are used in practice. Figure 4 quantifies the use within the operation groups and per metamodel version (cf. Table III). An operation type is in the set of used operations if there is at least one exemplar

[¶]**Note**: Except for the variants *V-Modell XT 1.3* and *V-Modell XT Bund 1.0*, for confidentiality reasons, we are not allowed to relate the findings to a variant from Figure 2; we provide the data, but anonymized. A collection of publicly available material is provided by the Weit e.V.: `www.weit-verein.de/varianten.html` (in German).

[‖]An operation group comprises all operation types that are logically related (e.g., changes on work products).

DOI: 10.1002/smr

Table III. Used variability operations by variant, operation type, and metamodel release.

Operation Group	Variant Bund			Variant A			Variant B			Variant C		
	1.3	1.3B	∑	1.3	1.3B	∑	1.3	1.3B	∑	1.3	1.3B	∑
Discipline Variations	1	12	13	0	0	0	0	0	0	2	1	3
Work Product Variations	1	11	12	3	0	3	3	2	5	0	1	1
Topic Variations	2	19	21	5	0	5	9	0	9	1	0	1
Activity Variations	0	1	1	1	0	1	0	0	0	0	0	0
Task Variations	0	0	0	0	0	0	0	0	0	0	0	0
Role Variations	4	47	51	0	0	0	24	17	41	14	18	32
Tailoring Variations	0	4	4	1	0	1	1	0	1	0	0	0
Decision Gate Variations	0	10	10	3	0	3	1	0	1	0	0	0
Description Replacements	25	0	25	1	0	1	4	0	4	24	0	24
Description Add-ons	0	0	0	2	0	2	4	0	4	1	0	1
Description Re-Arragements	1	1	2	1	0	1	6	0	6	5	0	5
Description Removements	0	3	3	0	0	0	0	1	1	0	5	5
Tool/Method Ref. Variations	0	0	0	0	0	0	0	0	0	11	0	11
Mapping Variations	0	4	4	0	0	0	0	0	0	0	1	1
Appendix Variations	0	21	21	0	0	0	0	0	0	0	0	0
	34	133	**167**	17	0	**17**	52	20	**72**	58	26	**84**

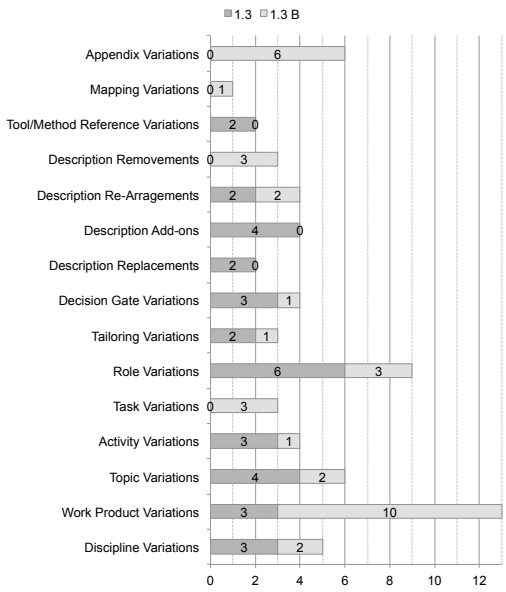

Figure 3. Defined variability operation types per metamodel version.

Figure 4. Used/unused variability operation types per metamodel version.

in any of the studied variants. Figure 4 shows which metamodel defines how many operation types (per operation group) and how many of them are used in the variants (overall count).

Table III gives the more detailed perspective based on the exemplars per operation group. The table does not list V-Modell XT variant "D" (Figure 2), as this variant does not contain any variability operation exemplars, i.e., this variant uses SPrL features, but does not use the variability operation instrument (cf. Section 4.4).

Unused Variability Operations Table IV lists the defined, but unused operation types. The metamodel "1.3" defines 34 variability operation types of which 12 remain unused (35.3%). The metamodel "1.3B" introduces 35 new variability operation types; 13 thereof remain unused (37.1%). In summary, across all metamodel versions, 25 out of 69 (36.2%) operation types are not used.

DOI: 10.1002/smr

Table IV. Unused variability operation types.

Operation Type	MM
AddDisciplineDescriptionPrefix	1.3
AddDisciplineDescriptionPostfix	1.3
DeleteWorkProduct	1.3B
ChangeWorkProduktDiscipline	1.3B
RenameCreatingDependency	1.3B
RenameTailoringDependency	1.3B
ReplaceTailoringDependencyDescription	1.3B
ArrangeSubTopic	1.3
AddActivityDescriptionPrefix	1.3
AddActivityDescriptionPostfix	1.3
RemoveTask	1.3B
RenameTask	1.3B
ReplaceTaskDescription	1.3B
RemoveResponsibility	1.3
AddRoleDescriptionPrefix	1.3
RefineRole	1.3
AddProcessModule	1.3
AddDecisionGateDescriptionPrefix	1.3
AddChapterTextPrefix	1.3
AddSectionTextPrefix	1.3
ChangeSectionNumber	1.3B
RemoveChapter	1.3B
RemoveGlossaryItem	1.3B
ReplaceGlossaryItemDescription	1.3B
RemoveAbbreviation	1.3B

Table V. Most frequently used variability operations.

Operation Type	QTY	MM
ReplaceSectionText	46	1.3
ChangeRoleClass	36	1.3B
ReplaceRoleDescription	34	1.3B
RenameRole	22	1.3
RemoveLiteratureReference	19	1.3B
RemoveTopicAssignment	16	1.3B
ChangeResponsibility	16	1.3
RemoveSupportingRole	12	1.3B
ArrangeSection	12	1.3
ChangeDisciplineNumber	11	1.3B

Most Frequently Used Variability Operations Several operation types are frequently used across several process variants. Table V lists the top-10 of the most frequently used operation types, including their number of exemplars and the metamodel that defines a particular operation type. The data shows that most of the frequently used operations modify text fragments of the process description (e.g., *ReplaceSectionText*, *ArrangeSection*) or alter the role model (e.g., *ReplaceRoleDescription*, *RenameRole*). Furthermore, the operations listed in Table V contribute 224 of the 340 investigated operation exemplars, i.e., the top-10 operation types contribute 65.9% of all found operation exemplars.

Interpretation As 25 operation types remain unused, one may conclude that about one third of the variability operations seems to be dispensable. The reason for the existence of such operations is mainly for process language completeness. For instance, the metamodel defines a pair of *Rename** and *Replace** operations for each of the process dependency types. The definition of these operations was a design decision during the development of the metamodel "1.3B". Since the V-Modell XT is designed as a generic framework, we cannot judge yet the relevance of the unused operation types, as future process variants may use them.

Our findings also show that the most frequently used operations address the customization of description texts, e.g., *ReplaceSectionText* or *ReplaceRoleDescription*. For instance, in one process variant, the entire introduction to the process model was "rewritten" to reflect the special requirements of the current context, e.g., new procedures for handling quality gates, support to conduct the tailoring for the project context, improved integration of the process variant with the company-wide quality management, and integration of agile practices and support tools (use of *ReplaceSectionText*[**]). Another process variant frequently used the operation *ReplaceRoleDescription* to replace the generic role profiles by those established in the particular

[**]Note: An operation exemplar may address small and local modifications as well as extensive modifications that impact the whole process. Furthermore, a particular process model element can be addressed by multiple variability operations.

DOI: 10.1002/smr

organization to align the customized process with the actual way of working. In the context of SPrLs, we interpret the frequent use as a standard use case in which a generic process description and a generic role model are refined for a particular process variant. Apart from just replacing text elements, the process's structure is also subject to change. For example, the operations *ChangeResponsibility* and *ArrangeSection* modify the process structure by modifying the responsibility for work products (work products are assigned to new/other roles) and, respectively, restructuring the process documentation (reordering existing chapters or injecting new ones). Furthermore, we found variability operations also applied to compensate metamodel evolution. For instance, the operation type *ChangeRoleClass*, which is on the second rank (Table V, 36 exemplars in two process variants), does not change any description text, but modifies the structure of role definitions in the process model in response to a metamodel evolution. In particular, this operation adds further metadata to "legacy" role elements that is required in newer versions of the metamodel. A variability operation is then used to reuse and update those role model elements without the need to change the reference model.

In summary, beside content-related variability operations, we also found variability operations modifying the structure of process assets, e.g., to enable for backward-compatibility. However, the analysis of variant "D" (no operation exemplars) also shows that variability operations are only one instrument among others and, thus, SPrLs can also be created and managed using other mechanisms.

4.4. RQ 3: Further Strategies to Provide Variability

The third research question aims at studying whether there are situations in which variability is required, but not implemented using the variability operation instrument. We found two process variants in which variability operations were not applied, but where other strategies to realize variability were used. Table VI provides a description of these variability strategies.

Table VI. Variability strategies not using the variability operation instrument.

Strategy	Description
Masking	Masking is, basically, no variability operation, although it could be considered as such. *Pre-tailoring* is a built-in mechanism that allows for a coarse-grained modification of configuration containers (e.g., to remove whole sub-processes). Furthermore, to construct a process variant, SPrLs also allow for adding (completely) new process assets. The combination of pre-tailoring and adding new content can be used in a strategy called "masking" that allows for *replacing* whole sub-processes. In the studied V-Modell XT variants, we found two cases in which masking was used to realize variability (top-level configurations, project type variants, were removed from the reference process and substituted by similar ones). *Rationale:* In several project type variants, existing process modules should be replaced by equivalent, but customized ones. However, no variability operation was defined to perform this replacement. Moreover, it turned out that a variability operation could not be implemented, as operation exemplars refer by *id* to the process assets to be modified. The used metamodel, however, did not provide these attributes for the specific model element referring the process modules in the process configuration. That is, the missing variability operation caused by a gap in the metamodel was "faked" using masking and, thus, is not traceable during an automatic compliance check anymore.
New Sub-Processes	The analysis of variant "D" showed a setting in which a variant was derived without using variability operations. As mentioned before, SPrLs also allow for deriving a process variant by adding new content. Variant "D" is an example: The reference model was just taken and *extended* by new content, e.g., new process modules, new roles, and new project type variants assembling the new process modules and such from the reference model. The new content showed no need for the use of variability operations as, for instance, no re-naming or text replacements were necessary.

In the *new sub-processes* strategy, new (sub-)processes were introduced. This strategy does not require the use of variability operations and, essentially, realizes 'true' extensions. However, both instruments can also be combined, e.g. in variant "B" and variant "C", new sub-processes were introduced, and variability operations were also used. Furthermore, in the variant "Bund", we found the *masking* strategy, which was used to compensate a technical gap in the metamodel. In this strategy, several other operations were used to mimic missing variability operations.

DOI: 10.1002/smr

Interpretation Variability operations are a meaningful tool. However, there are settings in which variability operations seem unnecessary. For instance, if a process variant mainly comprises added process assets while strictly adhering to a given reference process, variability operations may not be necessary (e.g., for the V-Modell XT variant "D"). Furthermore, there are settings in which variability operations cannot be defined due to metamodel-related gaps, but would be beneficial (e.g., in the V-Modell XT variant "Bund"). Settings using the masking strategy point to candidates for future metamodel improvements, such as adding *id* tags and defining appropriate new variability operations.

4.5. Validity of the Results

We evaluate our findings and critically review our study regarding the threats to validity. The *internal validity* could be threatened by a bias toward the variant construction process, because two of the authors are also the developers of the metamodels (and partially the processes). We minimized this threat by relying on an analysis tool, which was applied to all variants, and by calling in a third independent researcher. The *external validity* is threatened as we have little knowledge to what extent we can generalize our results, e.g., to other SPrLs, as there are no similar cases available so far. As our study provides the first analysis of variability in this context, a generalization of the findings is, so far, not our intention. Instead, we are interested in analyzing the feasibility of variability operations based on given case, and to prepare future research on SPrLs.

5. CONCLUSION

Our main goal was to analyze the feasibility of the variability operation instrument to support the systematic development of software process lines, and to develop an initial catalog of practically used variability operations. To this end, we studied the V-Modell XT SPrL, and analyzed the reference process and 5 variants using the built-in SPrL features. So far, we found two metamodel versions defining variability operations: the metamodel of the reference process defines 34 variability operation types, and the enhanced metamodel of the V-Modell XT Bund adds 35 more types. In summary, we found 69 variability operation types, which allow process engineers to declaratively modify process content (e.g., by providing new text snippets) and to modify the structure of a process (e.g., by changing responsibilities, removing references, and modifying the tailoring behavior). Furthermore, we investigated which variability operations were applied in practice, which allows us to rate the feasibility of this instrument. Among the 69 identified operation types, we found that 25 where defined, but remained unused. Unused operation types were either defined during the initial development of this instrument (based on experiences or to allow for a constrictive compliance), or the operations were defined during metamodel improvements (mainly to improve the completeness of the process language). Our findings also show that the variability operation instrument also serves metamodel evolution, which inherently happens in long-term SPrL development. In particular, we found variability operations that allow for structurally modifying 'legacy' process assets for reuse in newer versions of a process. Finally, we found settings in which variability operations were not used, as they were either unnecessary or missing. If variability operations were missing, we found a work around used to mimic missing operation types and, thus, we could also identify metamodel improvement candidates.

In summary, we found the variability operation instrument sufficient to support process engineers in constructing a (new) process variant from an SPrL. However, variability operations are only one instrument among others and they can be combined with other instruments. We also showed the difficulty to define meaningful variability operations as we found a number of variability operations defined, but unused. Still, we could find for all these operations a rationale for why they are part of the model whereby further evaluation needs to remain in scope of future investigations.

Practitioners can benefit from our findings. As variability operations are a means to declaratively define modifications of a reference process, this concept offers payoffs in domains in which

DOI: 10.1002/smr

regularized processes must be applied, e.g., in medicine, automotive, and in avionics. A company-specific process can declare the modifications regarding the reference process using variability operations that can be easily tracked and, thus, support audits and assessments. Furthermore, using variability operations helps to reduce process maintenance cost. In the optimal case, the company-specific process can be updated by replacing the old reference model by a newer version and by executing the merge procedure again. Furthermore, due to the separation of the reference- and the extension model, companies can establish a process life cycle management independent from the reference model. That is, as a company is in full control of its own variant, it can follow its own maintenance and improvement cycles, and, furthermore, can decide which of potentially refreshed reference contents is adopted for the company's variant(s).

Relation to Existing Evidence In [2, 3, 18–20], initial research was done in the area of evolution-driven variability analysis. However, these contributions aim at identifying variations in given models. Our research is focussed on a constructive approach that supports variability by design. Martínez-Ruiz et al. [15] conducted a study in which they investigated the constructors used in tailoring, revealing that current tailoring constructors do not meet industry requirements. They argue for instruments allowing for variability and consistency at the same time. Carvalho et al. [7] provide a systematic literature review of SPrLs and their implementation in practice, and conclude that SPrL concepts still have to be considered immature due to a lack of reported empirical evidence.

Due to its dissemination, SPEM is a candidate for sophisticated variability concepts, and as we found in [10], a number of improvement proposals are already available. SPEM defines a set of basic variability operations (e.g., extends or replaces), which is the basis for several SPrL-related improvement proposals (see, e.g., [13, 14, 16, 21]). However, no case study in the context of SPEM is yet available presenting concrete practical experiences.

Limitations The major limitation is that our study is based on the V-Modell XT only. However, it is the only process framework that provides this explicit support to create SPrLs. Furthermore, directly comparing the V-Modell XT framework and SPEM, significant differences regarding the notion of process variability as well as process tailoring become evident. While process customization (at the company level) and process tailoring (at the project level) are clearly separated processes in the V-Modell XT, SPEM does not provide such a clear differentiation. Therefore, as the concepts as well as the processes differ, transfer and generalization of the findings have to be prepared carefully. Furthermore, the V-Modell XT provides a rich portfolio of instruments to create process variants. In this article, we focused on the variability operation instrument, and we barely scratched the surface regarding other instruments (e.g., Section 4.4).

Future Work As our investigation is based on a snapshot of the V-Modell XT process line, which is based on the version 1.3 of the reference model and all related variants, the study at hand needs to be repeated for future versions of the reference model and its variants. A repeated analysis allows for analyzing the evolution of the instrument over time, e.g., regarding the question whether there are new variability operations (e.g., addressing the gaps discussed in Section 4.4), or whether unused variability operations are removed or used in future versions. Such an analysis can be accompanied by an in-depth analysis investigating which elements are modified in the different variants and if there are variability pattern across the different variants, which can be used to derive particular improvement requirements for the reference model.

As a second step, independent research is necessary to analyze the transfer options to other frameworks. Variability operations are a meaningful instrument to support process variability, however, as we already discussed in [8] and as also mentioned in [15], there is a gap in process frameworks regarding the capability to model flexible processes. This gap needs to be closed and, thus, it needs to be further investigated whether variability operations can contribute.

REFERENCES

DOI: 10.1002/smr

1. J. A. H. Alegría and M. C. Bastarrica. Building software process lines with casper. In *International Conference on Software and Systems Process*, pages 170–179. IEEE, 2012.
2. O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping software process models: Initial concepts and experience from defining space standards. In *International Conference on Software Process*, volume 5007 of *Lecture Notes in Computer Science*, pages 160–172. Springer-Verlag Berlin Heidelberg, 2008.
3. O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping Software Process Lines. *Software Process: Improvement and Practice*, 14(3):181–197, 2009.
4. G. Chastek, P. Donohoe, K. C. Kang, and S. Thiel. Product line analysis: A practical introduction. Technical report, Software Engineering Institute, 2001.
5. L. Chen, M. A. Babar, and N. Ali. Variability management in software product lines: A systematic review. In *International Software Product Line Conference*, pages 81–90. Carnegie Mellon University, 2009.
6. S. Cohen. Guidelines for developing a product line concept of operations. Technical Report CMU/SEI-99-TR-008, SEI, 1999.
7. D. D. de Carvalho, L. F. Chagas, A. M. Lima, and C. A. L. Reis. Software process lines: A systematic literature review. In *Software Process Improvement and Capability Determination*, volume 477 of *Communications in Computer and Information Science*, pages 118–130. Springer International Publishing Switzerland, 2014.
8. G. Kalus and M. Kuhrmann. Criteria for software process tailoring: A systematic review. In *International Conference on Software and Systems Process*, pages 171–180. ACM Press, 2013.
9. M. Kuhrmann. *Konstruktion modularer Vorgehensmodelle*. PhD thesis, Technische Universität München, 2008.
10. M. Kuhrmann, D. M. Fernandez, and R. Steenweg. Systematic software process development: Where do we stand today? In *International Conference on Software and Systems Process*, pages 166–170. ACM Press, 2013.
11. M. Kuhrmann, D. M. Fernández, and T. Ternité. Full set of analyzed variability operations. Online: http://goo.gl/OQvmDk, 2014.
12. M. Kuhrmann, D. M. Fernández, and T. Ternité. Realizing software process lines: Insights and experiences. In *International Conference on Software and System Process*, pages 99–108. ACM Press, 2014.
13. T. Martínez-Ruiz, F. García, and M. Piattini. Towards a spem v2.0 extension to define process lines variability mechanisms. In *Software Engineering Research, Management and Applications*, volume 150 of *Studies in Computational Intelligence*, pages 115–130. Springer, 2008.
14. T. Martínez-Ruiz, F. García, M. Piattini, and F. De Lucas-Consuegra. Process variability management in global software development: A case study. In *International Conference on Software and Systems Process*, pages 46–55. ACM Press, 2013.
15. T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini. Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal*, 20(1):229–260, 2012.
16. Martinez-Ruiz, T., Garcia, F., Piattini, M., and Münch, J. Modeling Software Process Variability: An Empirical Study. *IET Software*, 5(2), 2011.
17. M. Niazi and S. Zahran. *Industrial Engineering: Concepts, Methodologies, Tools, and Applications*, chapter Software Process Lines: A Step towards Software Industrialization, pages 1988–2002. Number 107. IGI Global, 2012.
18. A. Ocampo and J. Münch. Rationale modeling for software process evolution. *Software Process: Improvement and Practice*, 14(2):85–105, 2009.
19. A. Ocampo, J. Münch, and W. Riddle. Incrementally introducing process model rationale support in an organization. In *International Conference on Software Process*, volume 5543 of *Lecture Notes in Computer Science*, pages 330–341. Springer-Verlag Berlin Heidelberg, 2009.
20. A. Ocampo and M. Soto. Connecting the rationale for changes to the evolution of a process. In *International Conference on Product-Focused Software Process Improvement*, volume 4589 of *Lecture Notes in Computer Science*, pages 160–174. Springer-Verlag Berlin Heidelberg, 2007.
21. E. A. Oliveira, M. G. Pazin, I. M. S. Gimenes, U. Kulesza, and F. A. Aleixo. SMartySPEM: a SPEM-based approach for variability management in software process lines. In *International Conference on Product-Focused Software Process Improvement*, volume 7983 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag Berlin Heidelberg, 2013.
22. OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. Technical report, Object Management Group, 2008.
23. D. Rombach. Integrated software process and product lines. In *International Software Process Workshop (SPW)*, volume 3840 of *Lecture Notes in Computer Science*, pages 83–90. Springer-Verlag Berlin Heidelberg, 2005.
24. J. Schramm, P. Dohrmann, A. Rausch, and T. Ternité. Process model engineering lifecycle: Holistic concept proposal and systematic literature review. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 127–130. IEEE, 2014.
25. SEI. Software Product Lines. Online: http://www.sei.cmu.edu/productlines.
26. T. Ternité. Process lines: A product line approach designed for process model development. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 173–180. IEEE, 2009.
27. T. Ternité. *Variability of Development Models*. PhD thesis, Technische Universität Clausthal, 2010.
28. T. Ternité and M. Kuhrmann. Das V-Modell XT 1.3 Metamodell. Research Report TUM-I0905, Technische Universität München, 2009.

DOI: 10.1002/smr